Car MSRP

四技資管四甲
10446011 張紘銘

摘要

運用一學期所學的機器學習與深度學習，對 Kaggle 平台上所選的資料進行分析。

介紹

於 Kaggle 平台選擇 Car Features and MSRP 資料集，一台車的 MSRP（建議售價）與其特徵息息相關，平價親民的轎車與高價豪華的房車皆有影響 MSRP 的部分。

因此想藉由此資料集來分析影響 MSRP 的特徵。

資料集介紹

使用 Car Features and MSRP 資料集，包含

1. Make（製造商）
2. Model（車型）
3. Year（年份）、Engine Fuel Type（引擎油料類型）
4. Engine HP（引擎馬力）
5. Engine Cylinders（引擎汽缸）
6. Transmission Type（變速類型）
7. Driven_Wheels（傳動）
8. Number of Doors（車門數）
9. Market Category（市場標籤）
10. Vehicle Size（車輛大小）
11. Vehicle Style（車輛類型）
12. highway MPG（高速油耗）
13. city mpg（市區油耗）
14. Popularity（熱門度）
15. MSRP（建議售價）

資料集來源：https://www.kaggle.com/CooperUnion/cardataset

資料預處理

首先先將資料集匯入並查看資料

In [21]:
```
#import
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

%matplotlib inline
plt.style.use('ggplot')

df= pd.read_csv('data.csv')
df.sample(5)
```

Out[21]:

	Make	Model	Year	Engine Fuel Type	Engine HP	Engine Cylinders	Transmission Type	Driven_Wheels	Number of Doors	Market Category	Vehicle Size	Vehicle Style	highway MPG
7696	Toyota	Prius	2015	regular unleaded	134.0	4.0	AUTOMATIC	front wheel drive	4.0	Hatchback,Hybrid	Compact	4dr Hatchback	48
2889	Bentley	Continental Supersports	2011	flex-fuel (premium unleaded required/E85)	621.0	12.0	AUTOMATIC	all wheel drive	2.0	Exotic,Flex Fuel,Factory Tuner,Luxury,High-Per...	Midsize	Coupe	19
11524	BMW	X6 M	2016	premium unleaded (required)	567.0	8.0	AUTOMATIC	all wheel drive	4.0	Crossover,Factory Tuner,Luxury,High-Performance	Midsize	4dr SUV	19
4074	Chevrolet	Equinox	2017	regular unleaded	182.0	4.0	AUTOMATIC	front wheel drive	4.0	Crossover	Compact	4dr SUV	32
323	Nissan	370Z	2015	premium unleaded (required)	350.0	6.0	AUTOMATIC	rear wheel drive	2.0	Factory Tuner,High-Performance	Compact	Coupe	26

將高速與市區油耗計算平均油耗

In [22]:
```
#average MPG
for i in range(len(df)):
    average_MPG = (df.loc[i, 'highway MPG'] + df.loc[i, 'city mpg']) / 2
    df.loc[i, 'average MPG'] = average_MPG
df.sample(5)
```

Out[22]:

	Engine HP	Engine Cylinders	Transmission Type	Driven_Wheels	Number of Doors	Market Category	Vehicle Size	Vehicle Style	highway MPG	city mpg	Popularity	MSRP	average MPG
	134.0	4.0	AUTOMATIC	front wheel drive	4.0	Hatchback,Hybrid	Compact	4dr Hatchback	48	51	2031	30005	49.5
	199.0	8.0	MANUAL	four wheel drive	2.0	NaN	Midsize	2dr SUV	16	13	5657	2996	14.5
	420.0	6.0	AUTOMATED_MANUAL	all wheel drive	4.0	Luxury,High-Performance	Large	Sedan	26	17	1715	125600	21.5
	275.0	8.0	AUTOMATIC	front wheel drive	4.0	Luxury	Large	Sedan	23	15	61	38790	19.0
	335.0	6.0	AUTOMATIC	rear wheel drive	4.0	Luxury,Performance,Hybrid	Large	Sedan	30	23	3916	62100	26.5

檢查遺失資料

In [23]:
```
#missing data
total = df.isnull().sum().sort_values(ascending = False)
percent = (df.isnull().sum() / df.isnull().count()).sort_values(ascending = False)
missing_data = pd.concat([total, percent], axis = 1, keys = ['Total', 'Percent'])
missing_data
```

Out[23]:

	Total	Percent
Market Category	3742	0.314084
Engine HP	69	0.005792
Engine Cylinders	30	0.002518
Number of Doors	6	0.000504
Engine Fuel Type	3	0.000252
Driven_Wheels	0	0.000000
Model	0	0.000000
Year	0	0.000000
Transmission Type	0	0.000000
average MPG	0	0.000000
MSRP	0	0.000000
Vehicle Size	0	0.000000
Vehicle Style	0	0.000000
highway MPG	0	0.000000
city mpg	0	0.000000
Popularity	0	0.000000
Make	0	0.000000

填補缺失的市場標籤為 non，其餘缺失資料則刪除整列

In [24]:
```
#fill&drop
df = df.drop(['highway MPG', 'city mpg'], axis = 1).reset_index(drop=True)
df['Market Category'] = df['Market Category'].fillna('non')
df = df.dropna(axis = 0).reset_index(drop=True)
df.sample(5)
```

Out[24]:

	Make	Model	Year	Engine Fuel Type	Engine HP	Engine Cylinders	Transmission Type	Driven_Wheels	Number of Doors	Market Category	Vehicle Size	Vehicle Style	Popularity	MSR
2484	Dodge	Charger	2016	premium unleaded (recommended)	707.0	8.0	AUTOMATIC	rear wheel drive	4.0	Factory Tuner,High-Performance	Large	Sedan	1851	6594
10195	Ford	Taurus	2016	premium unleaded (recommended)	365.0	6.0	AUTOMATIC	all wheel drive	4.0	Factory Tuner,High-Performance	Large	Sedan	5657	4027
2785	Chevrolet	Colorado	2015	regular unleaded	305.0	6.0	AUTOMATIC	four wheel drive	4.0	non	Compact	Crew Cab Pickup	1385	2970
8378	Buick	Regal	2016	premium unleaded (recommended)	259.0	4.0	AUTOMATIC	all wheel drive	4.0	Performance	Midsize	Sedan	155	3399
9158	Toyota	Sienna	2017	regular unleaded	296.0	6.0	AUTOMATIC	all wheel drive	4.0	non	Large	Passenger Minivan	2031	4731

再次檢查遺失資料

In [25]:
```
#missing data
total = df.isnull().sum().sort_values(ascending = False)
percent = (df.isnull().sum() / df.isnull().count()).sort_values(ascending = False)
missing_data = pd.concat([total, percent], axis = 1, keys = ['Total', 'Percent'])
missing_data
```

Out[25]:

	Total	Percent
average MPG	0	0.0
MSRP	0	0.0
Popularity	0	0.0
Vehicle Style	0	0.0
Vehicle Size	0	0.0
Market Category	0	0.0
Number of Doors	0	0.0
Driven_Wheels	0	0.0
Transmission Type	0	0.0
Engine Cylinders	0	0.0
Engine HP	0	0.0
Engine Fuel Type	0	0.0
Year	0	0.0
Model	0	0.0
Make	0	0.0

one hot encoding

In [27]:
```
#one hot encoding
X = pd.get_dummies(df, dummy_na = False, columns=['Make', 'Model', 'Engine Fuel Type', 'Transmission Type', 'Market Category', 'Dr
y = df[['MSRP']]
X.sample(5)
```

Out[27]:

	Year	Engine HP	Engine Cylinders	Number of Doors	Popularity	MSRP	average MPG	Make_Acura	Make_Alfa Romeo	Make_Aston Martin	...	Vehicle Style_Convertible	Vehicle Style_Convertible SUV	Vehic Style_Coup
9754	2017	185.0	4.0	4.0	1720	26700	24.5	0	0	0	...	0	0	
9048	2002	300.0	8.0	4.0	1624	49705	20.5	0	0	0	...	0	0	
4580	1997	220.0	8.0	2.0	5657	2862	14.5	0	0	0	...	0	0	
5409	2016	170.0	4.0	4.0	873	21625	30.5	0	0	0	...	0	0	
7377	1997	93.0	4.0	2.0	2031	2178	28.5	0	0	0	...	1	0	

研究結果及討論

使用整體學習的隨機森林來觀察每一個特徵的重要度

將資料切分為 30%測試與 70%訓練，接著進行特徵標準化提高資料的準確度

In [8]:
```
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.metrics import classification_report, make_scorer
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestRegressor

# Split dataset into training and testing
print('Splitting into training and testing...')
y_unraveled = np.ravel(y)
X_train, X_test, y_train, y_test = train_test_split(X, y_unraveled, test_size=0.3)
print("Done training best classifier.")

# 特徵標準化(Standardization)¶
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()   # initial
sc.fit(X_train)   # fit
X_train_std = sc.transform(X_train)   # 可得到標準化的訓練資料集
X_test_std = sc.transform(X_test)      # 可得到標準化的測試資料集
```

訓練隨機森林迴歸，產生分數、錯誤與預設

```
clf = RandomForestRegressor(n_estimators=100, max_features="sqrt");
# The gradient boosting classifier didnt finish running
# clf = GradientBoostingClassifier(n_estimators=5)
clf = clf.fit(X_train_std, y_train);
print("Done training best classifier.")

print('Calculating error...')
y_pred = clf.predict(X_test_std);
scores = cross_val_score(clf,X_test_std,y_test, cv = 5)
print()

print("Scores:")
print(scores);
print("Mean Squared Error")
print(np.mean((y_test - clf.predict(X_test))**2))
print("Mean absolute error:");
mean_error = sum(abs(y_test-y_pred))/len(y_test);
print(mean_error);
print("Mean percent error: ")
print(mean_error/np.mean(y_test))
print("R Square:")
print(clf.score(X_test_std,y_test))
print()

print("ypred:");
print(y_test);
print(y_pred);
```

```
Splitting into training and testing...
Done training best classifier.
Training classifier...
Done training best classifier.
Calculating error...

Scores:
[0.97838758 0.98492575 0.99225634 0.68891766 0.98697943]
Mean Squared Error
277001791134.665
Mean absolute error:
760.5014785553062
Mean percent error:
0.019360791245018886
R Square:
0.994911129424718

ypred:
[25300  2148 39390 ...  2506 37855 31990]
[25382.35  2579.5  39275.85 ...  2364.87 38052.95 32153.  ]
```

Kfold

```
sc.fit(X)

plot_learning_curve(clf, "RandomForestRegressor", sc.transform(X), y_unraveled, cv=cv, train_sizes=np.linspace(0.2, 1.0, 5))

# np.linspace(0.2, 1.0, 5)是指切成 0.2 , 0.4, 0.6 , 0.8, 1，中間四個間隔是均分。
# 所有的X共1288筆(X.count())  ，如果cv中的n_splits=4代表每次做KFold是 training data: test data = 3:1  (test data is 1288*(1/4))
# 所以總共可用的訓練資料是1288*(3/4) = 966
# 為了觀察訓練資料學習曲線和驗證曲線的變化，所以用例如np.linspace(0.2, 1, 5)的比例再切訓練資料
# 共做五次的KFold，每次用到的訓練資料量依序是：966*0.2、966*0.4、966*0.6、....966*1
# 每次KFold的測試資料量就是：966*0.2*(1/3)、966*0.4*(1/3)...
# 再計算每次Kfold中的score平均、標準差作圖
# 圖中顯示3份training 及 1份cross-validation的準確率
```

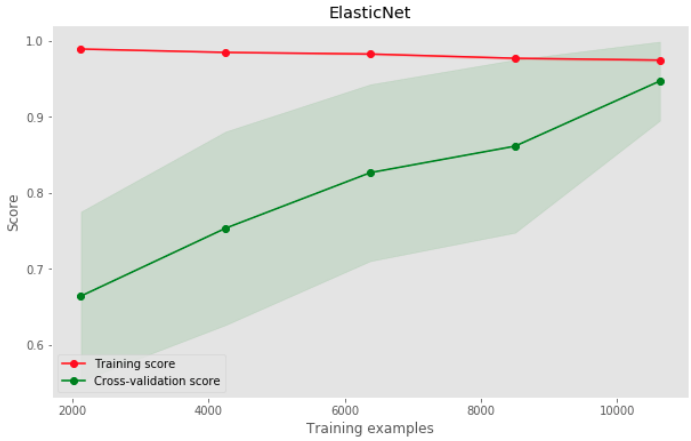
```
Automatically created module for IPython interactive environment
```

```
Out[9]: <module 'matplotlib.pyplot' from 'C:\\Users\\Jerry\\Anaconda3\\lib\\site-packages\\matplotlib\\pyplot.py'>
```

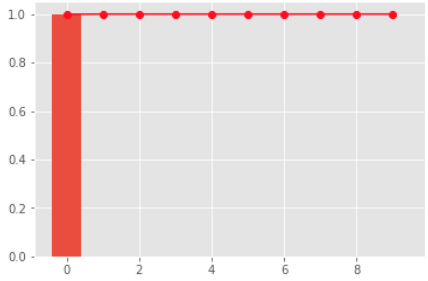
正規化

In [11]:
```
cv = KFold(n_splits=10, random_state=None, shuffle=True)
estimator = ElasticNet(alpha=1)

plot_learning_curve(estimator, "ElasticNet", sc.transform(X), y, cv=cv, train_sizes=np.linspace(0.2, 1.0, 5))
```

Out[11]:
```
<module 'matplotlib.pyplot' from 'C:\\Users\\Jerry\\Anaconda3\\lib\\site-packages\\matplotlib\\pyplot.py'>
```

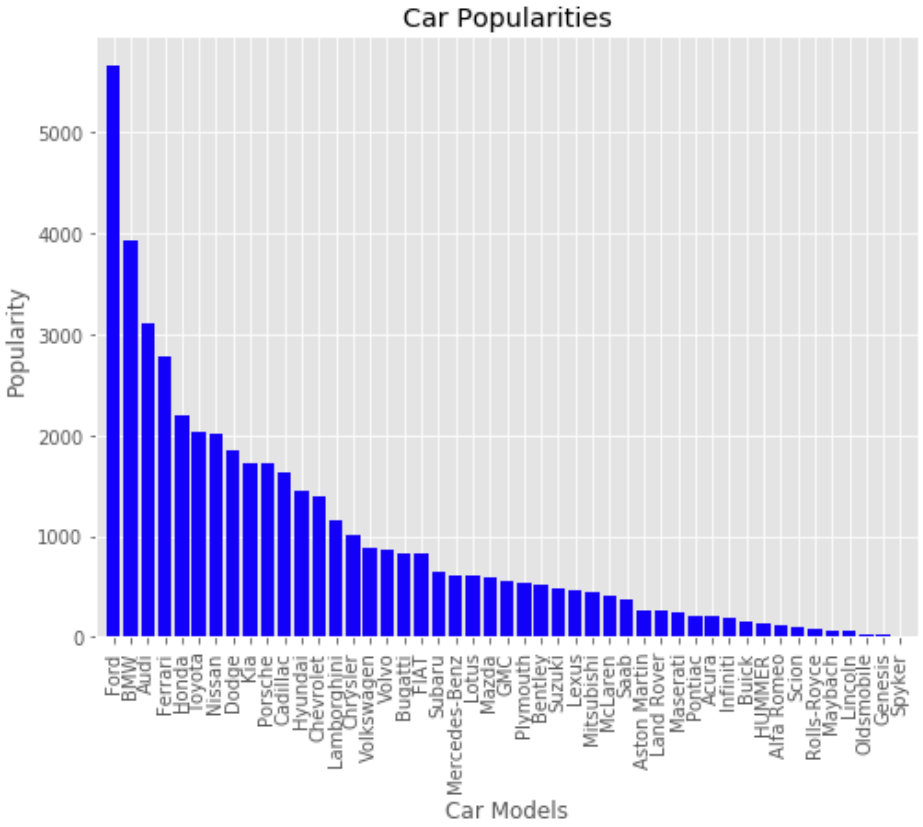


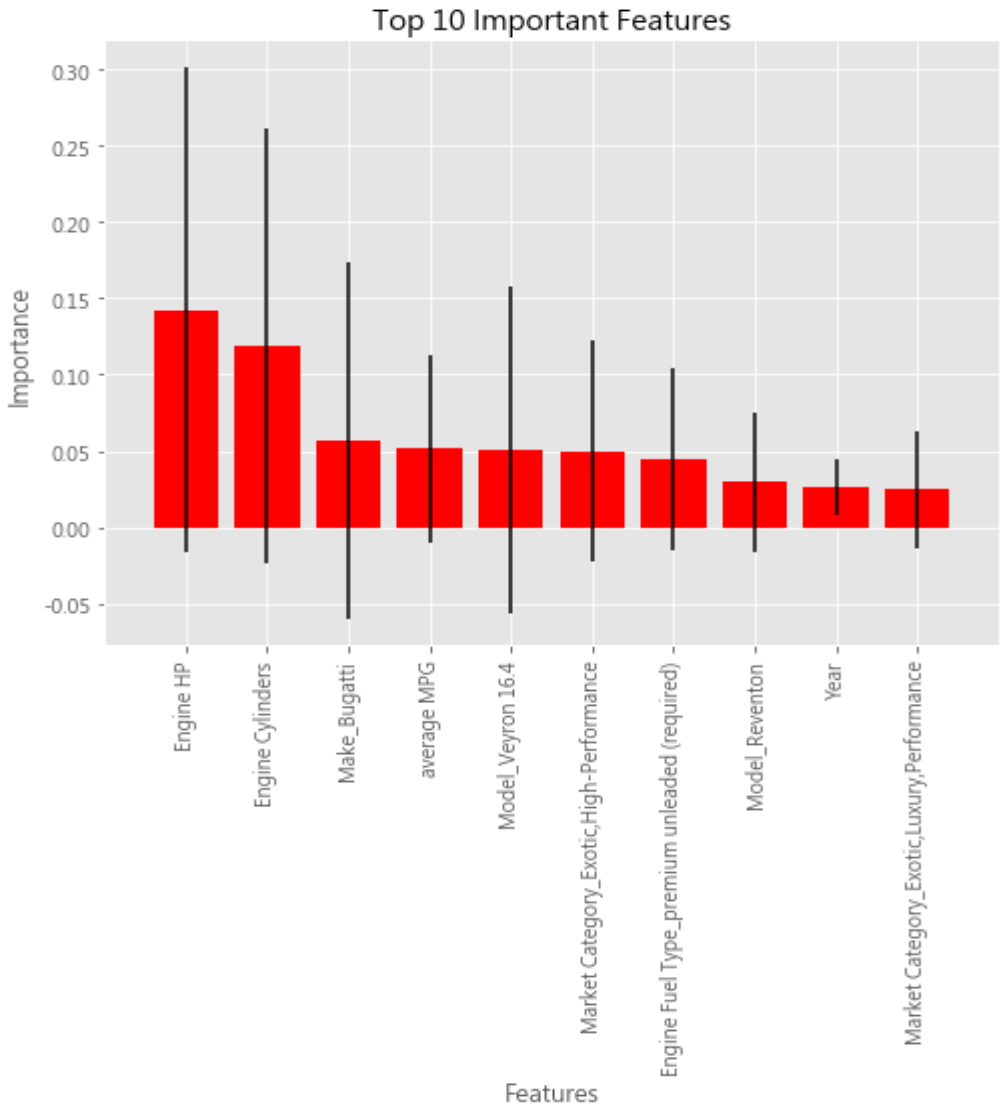
In [79]:
```
y1 = pca.explained_variance_ratio_
y2 = accumulate(pca.explained_variance_ratio_)
x = [i for i in range(len(y1))]

plt.figure()
plt.bar(x,y1,align='center')
plt.plot(x, y2, 'r-o')

plt.show()
```



發現在進行正規化後仍無法改善高變異問題，可能是因為資料量不足導致。

結論


Car Popularities
Popularity
0
1000
2000
3000
4000
5000
Ford
BMW
Audi
Ferrari
Honda
Toyota
Nissan
Dodge
Kia
Porsche
Cadillac
Hyundai
Chevrolet
Lamborghini
Chrysler
Volkswagen
Volvo
Bugatti
FIAT
Subaru
Mercedes-Benz
Lotus
Mazda
GMC
Plymouth
Bentley
Suzuki
Lexus
Mitsubishi
McLaren
Saab
Aston Martin
Land Rover
Maserati
Pontiac
Acura
Infiniti
Buick
HUMMER
Alfa Romeo
Scion
Rolls-Royce
Maybach
Lincoln
Oldsmobile
Genesis
Spyker
Car Models



分析之後發現影響建議售價最大的特徵為馬力，第二為引擎汽缸，馬力與引擎汽缸息息相關，第三則是品牌 Bugatti，與第五的車型 Veyron16.4 相關，是一間製造 Hypercar 的車廠，第四為平均油耗，高油耗通常為高級車或跑車，售價也高，第六與第十皆是進口高性能房車與跑車市場，第七是油料類型為高級無鉛汽油，第八為 Lamborghini 限量車 Reventón，第九則是年份。

影響特徵與熱門程度第一的品牌 Ford 結合，可以發現馬力、油耗、油料類型與年份影響價格最大。

參考文獻

https://www.kaggle.com